ЭЛЕМЕНТЫ ТЕХНИЧЕСКОГО ЗРЕНИЯ

СОПОСТАВЛЕНИЕ ИЗОБРАЖЕНИЙ. ТОЧЕЧНЫЕ ОСОБЕННОСТИ

СОПОСТАВЛЕНИЕ ИЗОБРАЖЕНИЙ

Для чего?

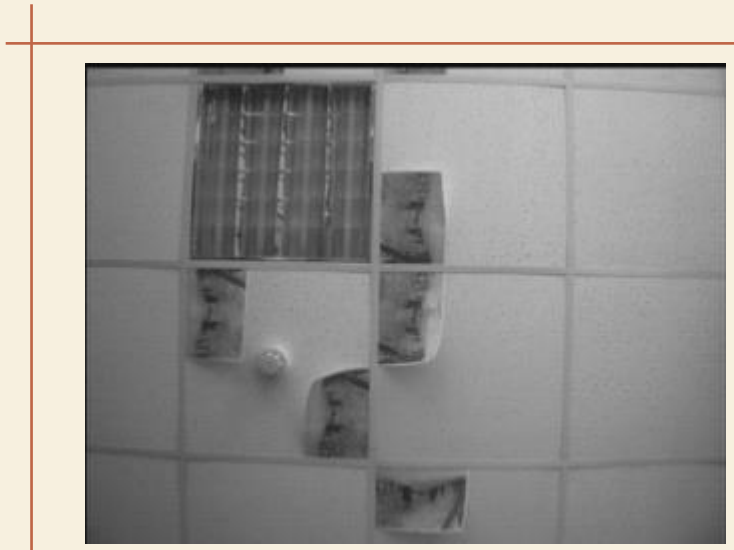
- Отслеживание объектов
- Стабилизация видео
- Генерация панорам
- 3х мерная реконструкция

СОПОСТАВЛЕНИЕ ИЗОБРАЖЕНИЙ

Два общих подхода:

- Согласование (alignment) на основе особенностей
  - Найти несколько соответствующих точек на обоих изображениях
  - Вычислить согласование
- Прямое(попиксельное) согласование
  - Поиск такого совмещения, при котором большинство пикселей совпадают

ПОПИКСЕЛЬНОЕ СОГЛАСОВАНИЕ



y

x

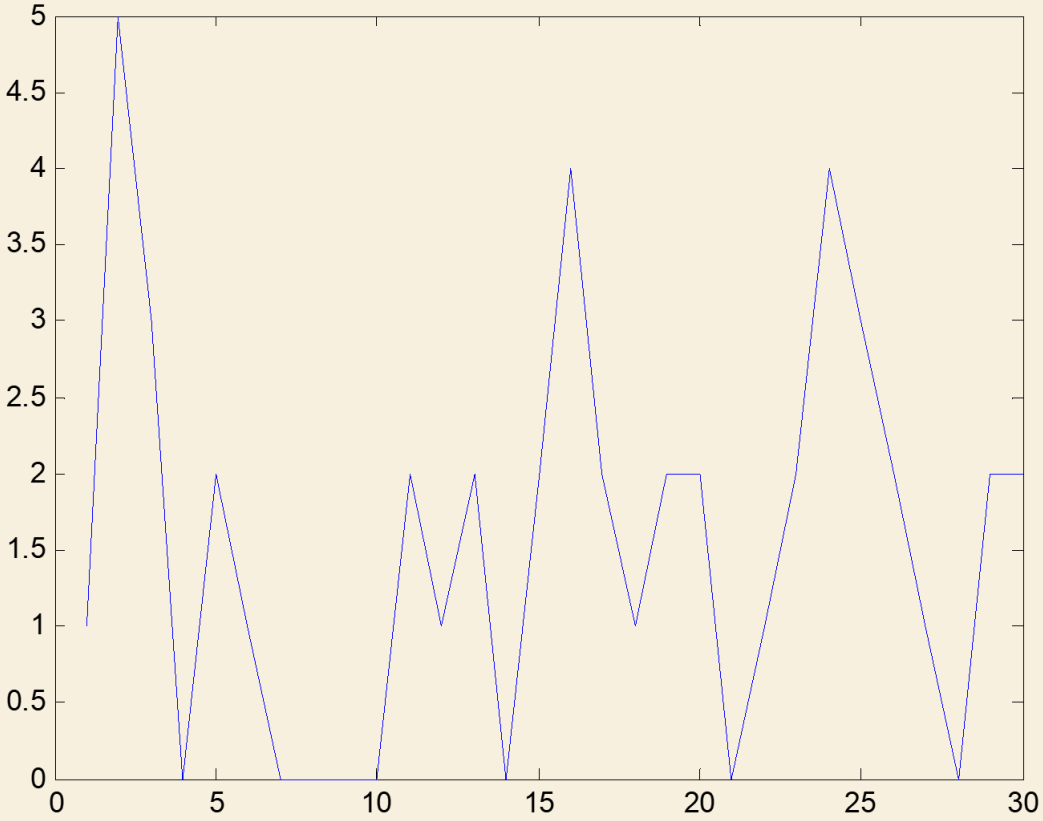
Одномерное движение по координате

ЧТО СРАВНИВАТЬ

- Выберем величину для сравнения, например яркость изображения в точке $I(x, y)$
- В качестве функции сравнения возьмём

$$F(\Delta x) = |I(x + \Delta x, y) - I(x, y)|$$

- Как найти его минимум?



ДЛЯ ОДНОЙ ТОЧКИ

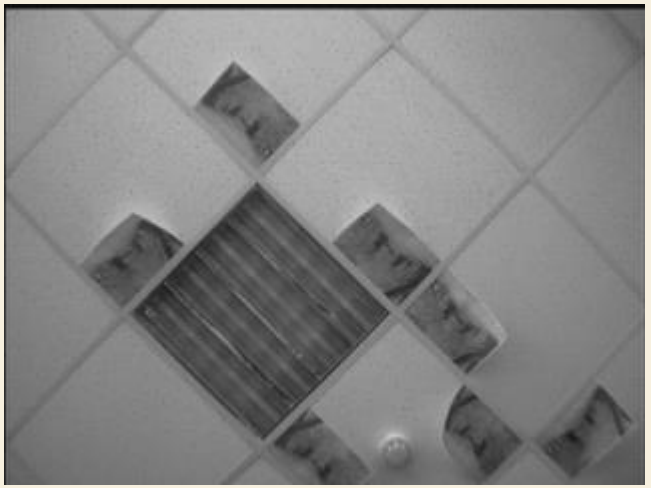
- Минимумов несколько
- Очевидно, что нужно использовать несколько точек

СУММАРНОЕ СОГЛАСОВАНИЕ

Для уточнения необходимо перебрать всё изображение

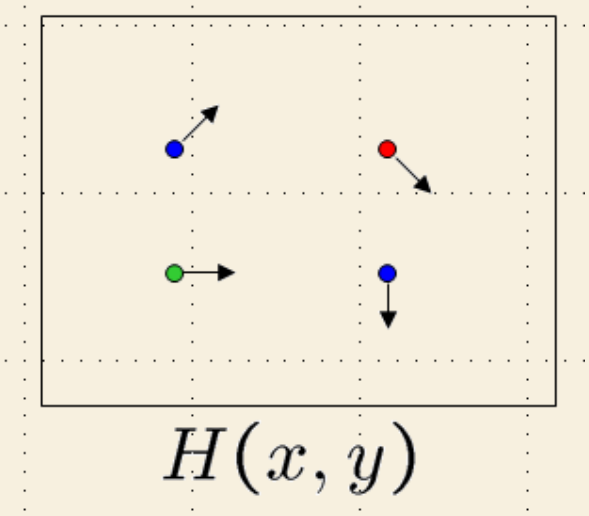
МНОГОМЕРНОЕ ДВИЖЕНИЕ

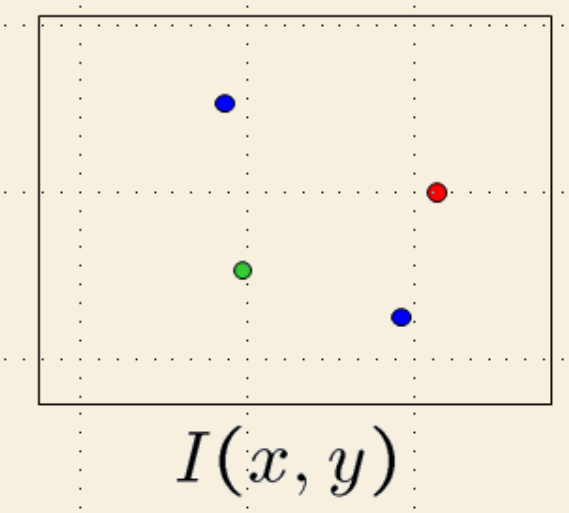
А что, если мы рассмотрим комбинированное движение параллельный перенос + поворот)?
Теперь нам придётся согласовывать 3 координаты одновременно!



Количество вычислений увеличивается! До $O(N^8)$

СОПОСТАВЛЕНИЕ ИЗОБРАЖЕНИЙ : ОПТИЧЕСКИЙ ПОТОК



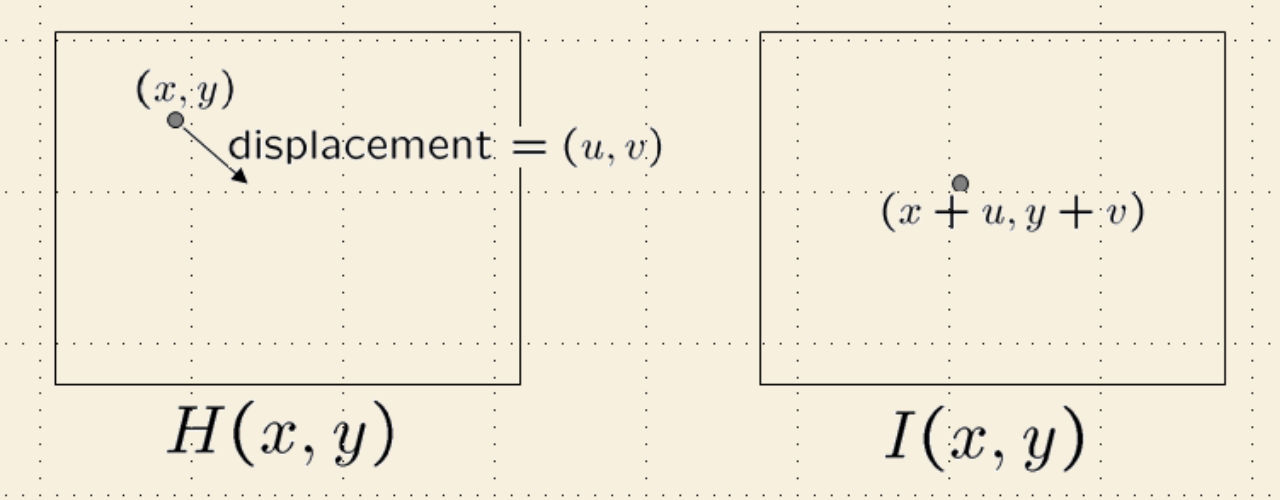


Ключевые предположения

- Константный цвет: точка в H выглядит также, как и в I
  - Для изображения в градациях серого, это постоянная яркость
- Малое движение: точки не уезжают далеко между кадрами

Эта задача называется поиск «оптического потока»

ОПТИЧЕСКИЙ ПОТОК



- Используем ограничения для формализации задачи
  - Постоянная яркость

$$I(x, y) - H(x, y) = 0$$

  - Малое смещение

$$I(x + u, y + v) = I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + O(u^2, v^2)$$

$$\approx I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v$$

ОПТИЧЕСКИЙ ПОТОК: УРАВНЕНИЯ В ТОЧКЕ

- Объединим ограничения:

$$0 = I(x + y, y + v) - H(x, y) \approx$$
$$\approx I(x, y) + I'_x u + I'_y v - H(x, y) \approx$$
$$\approx (I(x, y) - h(x, y)) + I'_x u + I'_y v \approx I'_t + I'_x u + I'_y v$$

- 1 уравнение, но 2 переменных (u,v) – как быть?

ОПТИЧЕСКИЙ ПОТОК: СИСТЕМА УРАВНЕНИЙ

- Идея: наложить дополнительные ограничения
  - Пусть оптический поток меняется плавно
  - пусть для всех пикселей из окрестности смещение постоянно! (u,v)
  - Тогда для окрестности 5x5 получим 25 уравнений!

$$\begin{bmatrix} I'_x(p_1) & I'_y(p_1) \\ I'_x(p_2) & I'_x(p_2) \\ \vdots & \vdots \\ I'_x(p_{25}) & I'_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I'_t(p_1) \\ I'_t(p_2) \\ \vdots \\ I'_t(p_{25}) \end{bmatrix}$$

СОПОСТАВЛЕНИЕ ИЗОБРАЖЕНИЙ : ОПТИЧЕСКИЙ ПОТОК

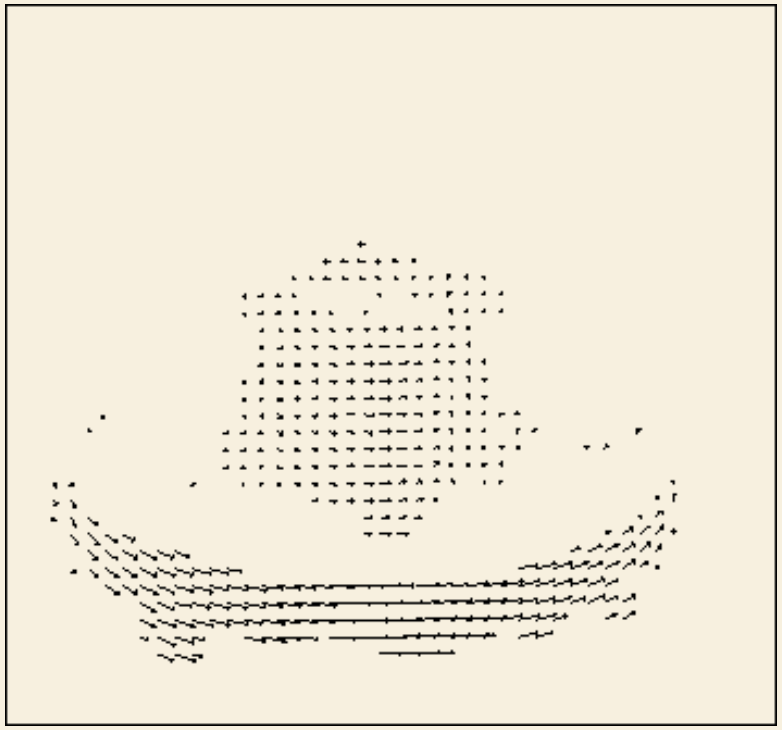
- Получаем задачу МНК

$$(A^T A)d = A^T b \quad => \quad d = (A^T A)^{-1} A^T b$$

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

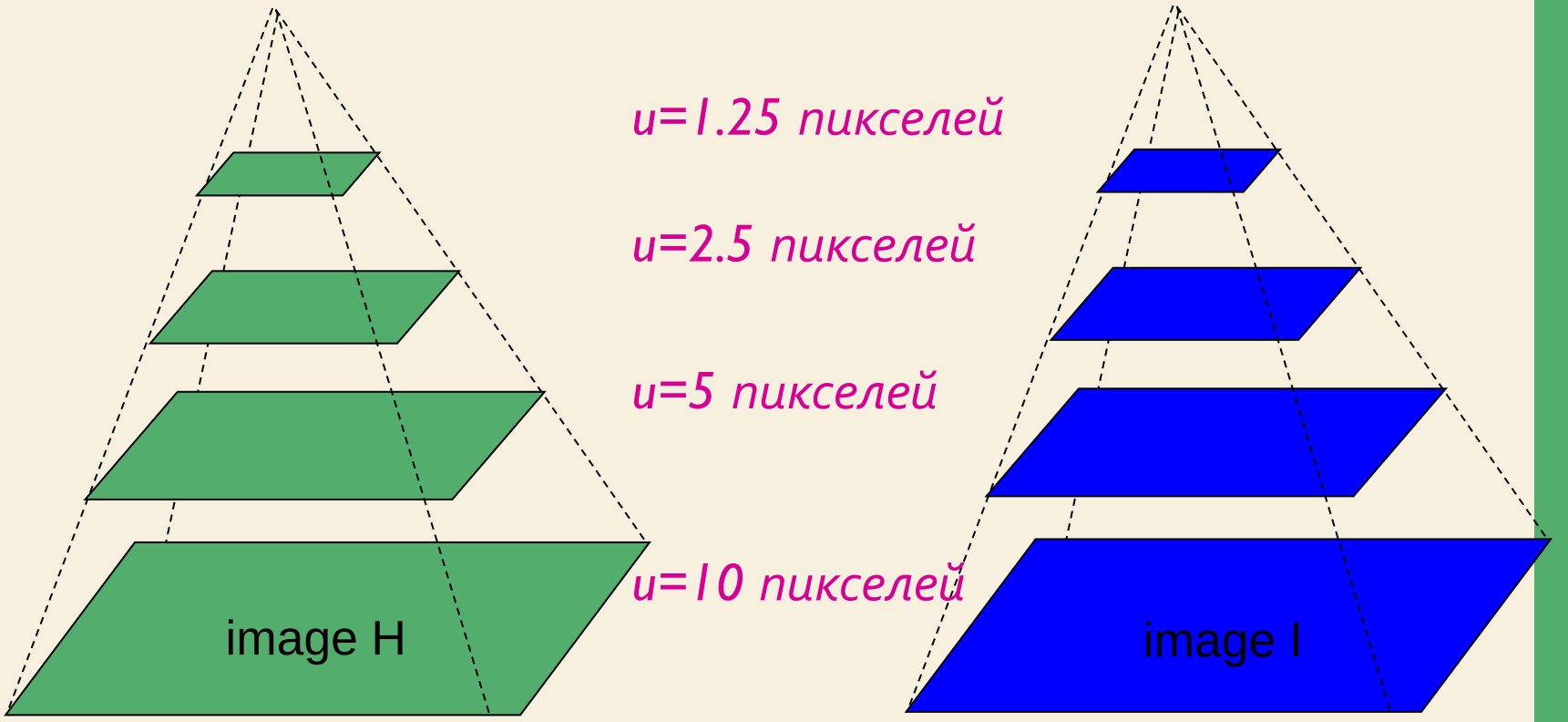
- Кроме того, можно наложить дополнительные ограничения, зная характер движения камеры

ОЦЕНКА ДВИЖЕНИЯ: ОПТИЧЕСКИЙ ПОТОК



Оценка движения начинается с оценки движения каждого пикселя
Затем рассматривается движение во всем изображении

ИЕРАРХИЧЕСКИЙ МЕТОД ОЦЕНКИ ДВИЖЕНИЯ

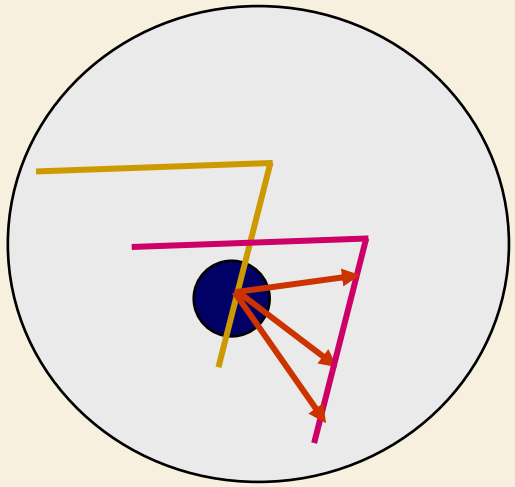


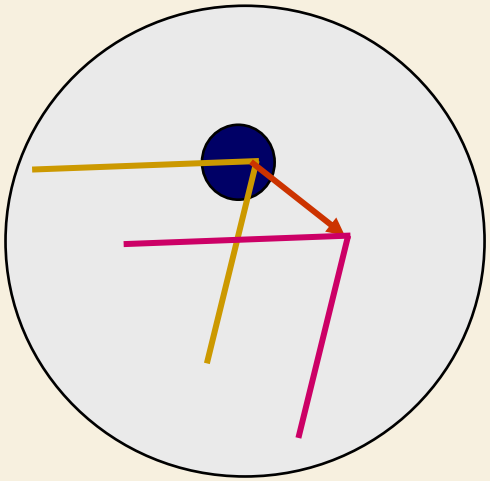
Гауссова пирамида для H

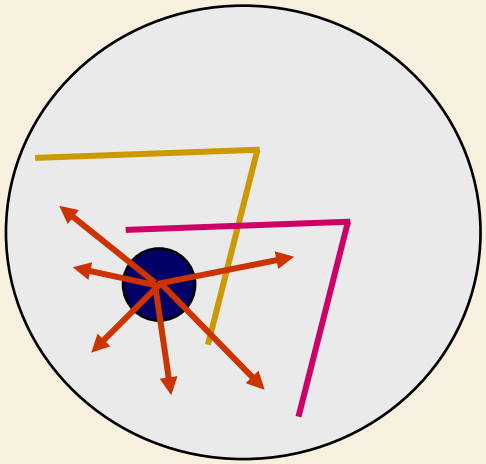
Гауссова пирамида для I

СОПОСТАВЛЕНИЕ ИЗОБРАЖЕНИЙ

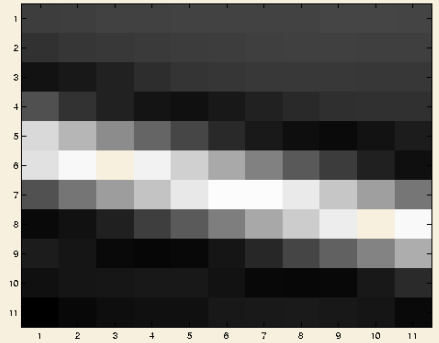
- Всегда ли задача разрешима?

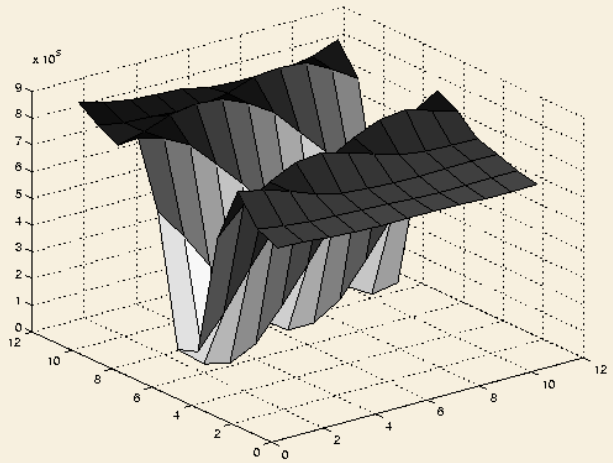






КРАЯ

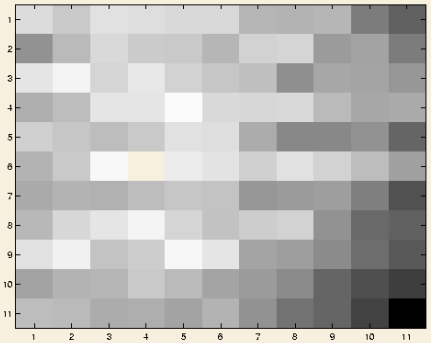


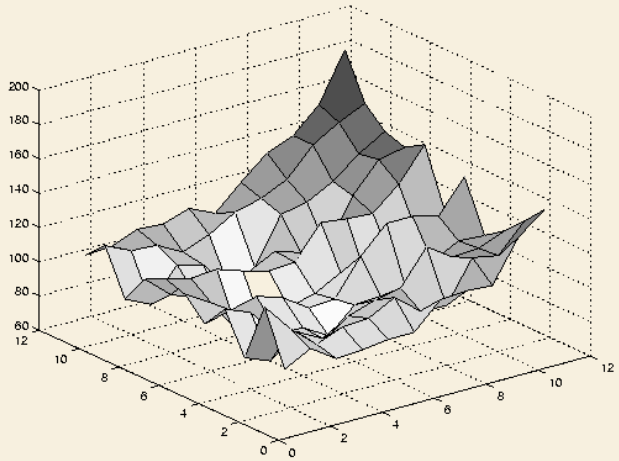


– большие градиенты

– большое λ_1, маленькое λ_2

СЛАБОКОНТРАСТНАЯ ТЕКСТУРА

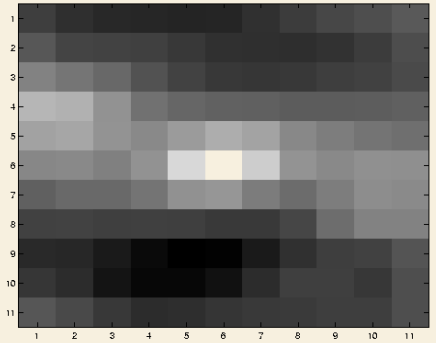


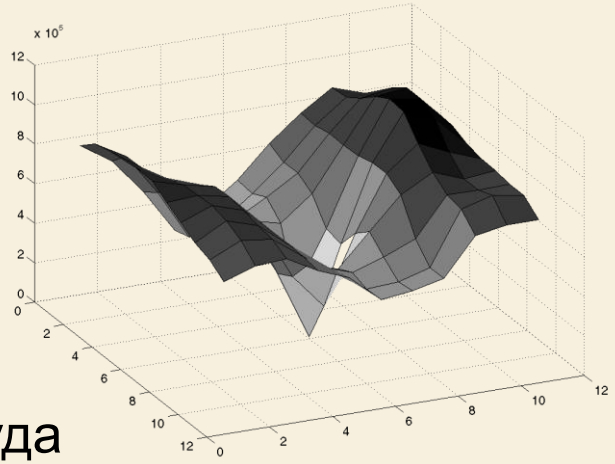


– величина градиента мала

– малое λ_1, малое λ_2

ТЕКСТУРИРОВАННАЯ ОБЛАСТЬ





– градиенты разные, большая амплитуда

– большое λ_1, большое λ_2

ДЕТЕКТОР ХАРРИСА

- Вычисляем собственные значения матрицы

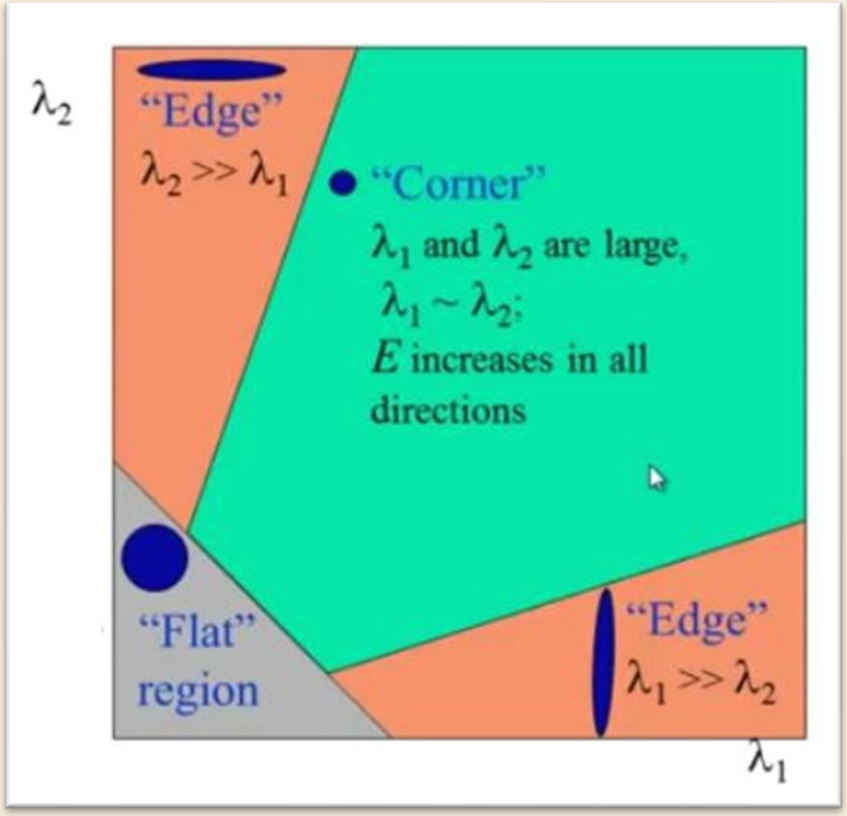
$$A^T A = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix}$$

- Вычисляем меру отклика

$$\mathrm{R} = \det A^T A - k \cdot tr A^T A$$

- Или

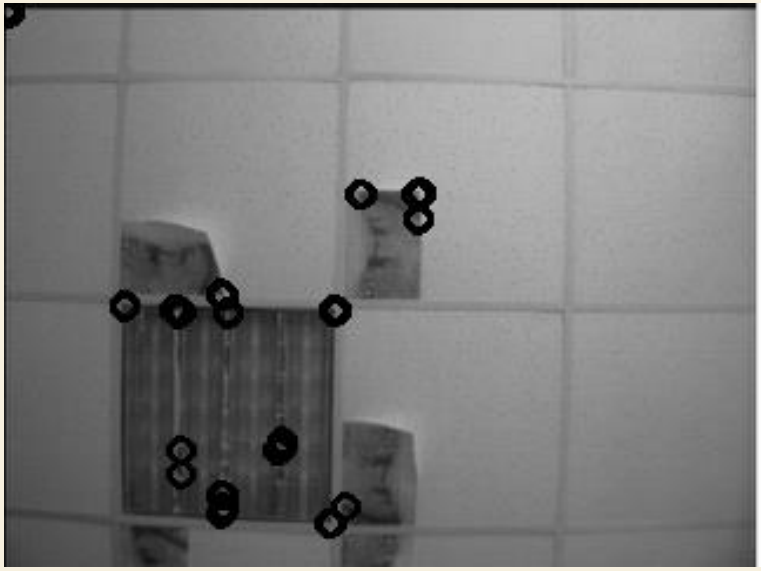
$$M = \frac{\det A^T A}{tr(A^T A)}$$



ДЕТЕКТОР ХАРРИСА

ИЗОБРАЖЕНИЕ

УГЛЫ



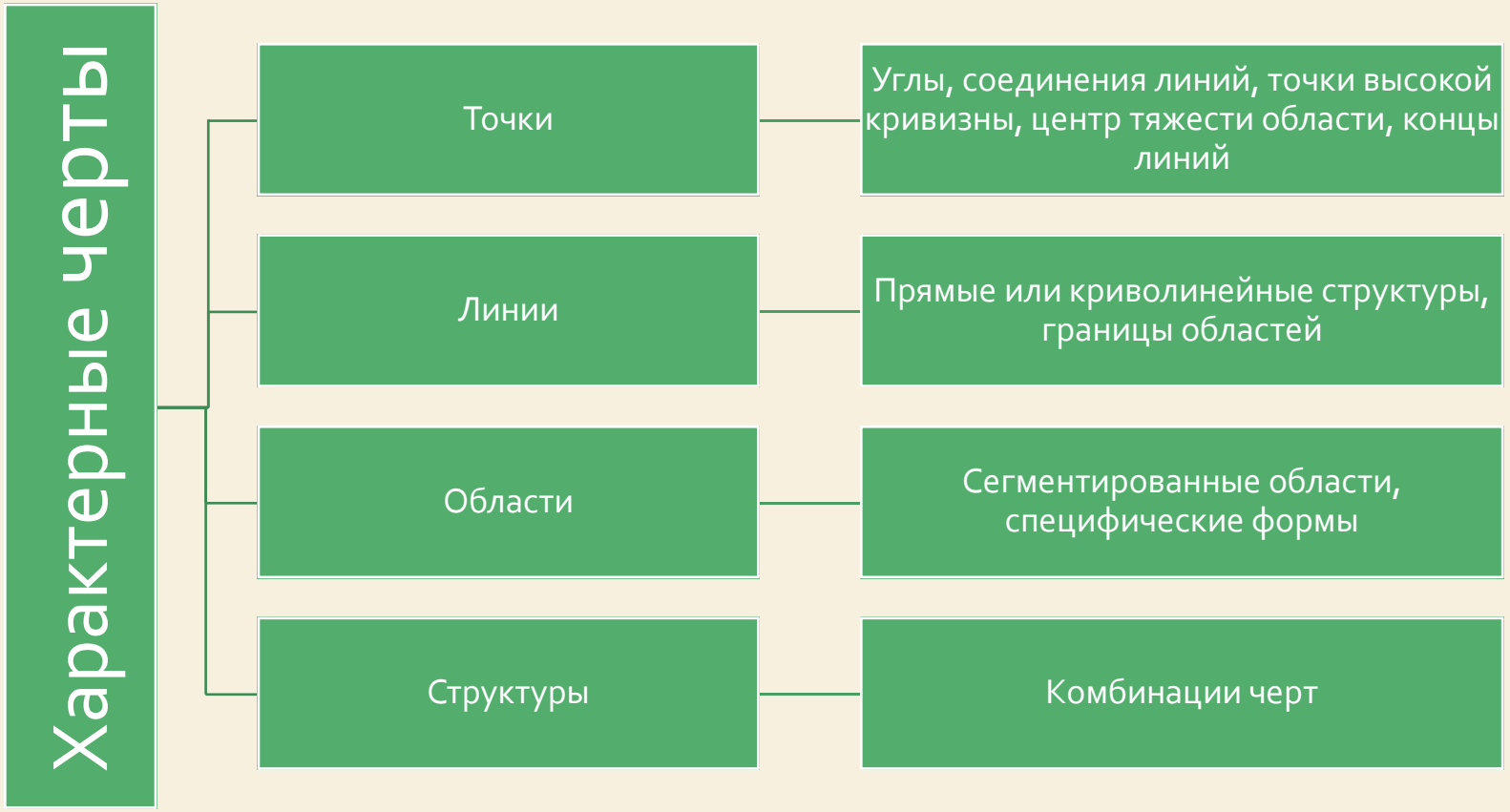
СОПОСТАВЛЕНИЕ ХАРАКТЕРНЫХ ЧЕРТ

- Мы будем использовать не всё изображение
- Сначала нам понадобится выделить «хорошие» объекты
- Мы будем сопоставлять их друг с другом
- Нам нужно какое-то описание таких объектов!

ЧТО ХОТЕЛОСЬ БЫ РАСПОЗНАВАТЬ?

ЧТО МЫ БУДЕМ РАСПОЗНАВАТЬ?

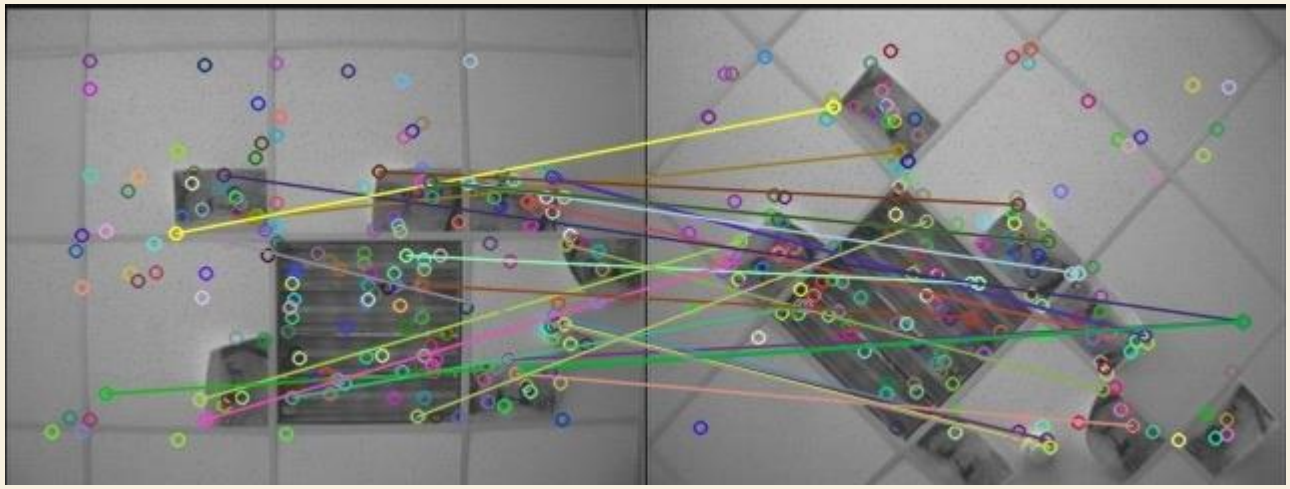
ХАРАКТЕРНЫЕ ЧЕРТЫ



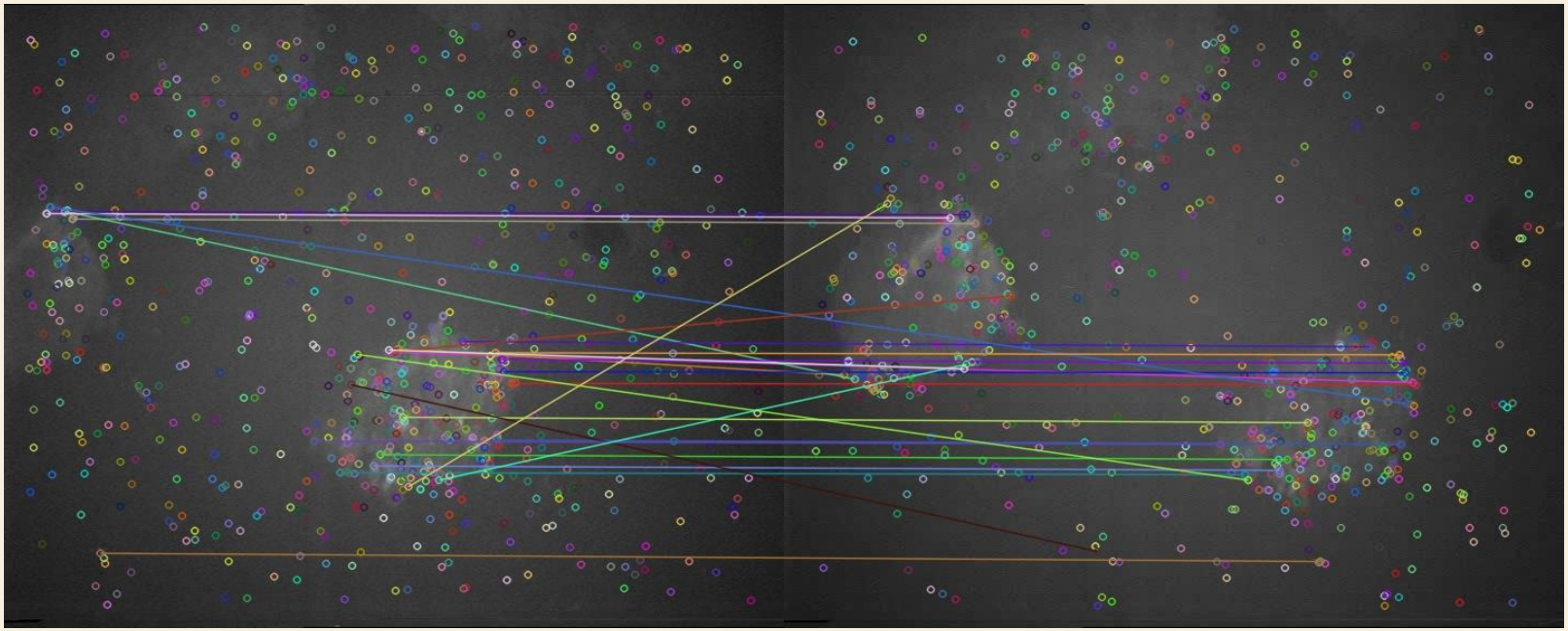
ХАРАКТЕРНЫЕ ЧЕРТЫ

Тип	лучше		хуже
Присутствие/плотность	точки	линии	области
Редкость/уникальность	области	линии	точки
Инвариантность	точки	линии	области
Устойчивость к шуму	области	линии	точки
Локализация	точки, особенно углы, центры	линии	области
Присутствие/плотность	точки	линии	области
Скорость	точки	линии	области
Влияние разрывов	области	линии	точки

ТОЧЕЧНЫЕ ОСОБЕННОСТИ



ТОЧЕЧНЫЕ ОСОБЕННОСТИ



ТОЧЕЧНЫЕ ОСОБЕННОСТИ

- Возможность выделить на последовательности кадров
- Уникальность
- Лёгкость выделения

Как этого достичь?

КЛЮЧЕВЫЕ (ОСОБЫЕ) ТОЧКИ

- Под ключевыми точками понимаются некоторые участки картинки, которые являются отличительными для данного изображения.
- Подобные точки каждый алгоритм определяет по своему.

СОСТАВЛЯЮЩИЕ ПРОЦЕССА ДЕТЕКТИРОВАНИЯ

- Детектор (feature detector) — осуществляет поиск ключевых точек на изображении.
- Дескриптор (descriptor extractor) — производит описание найденных ключевых точек, оценивая их позиции через описание окружающих областей.
- Матчер (matcher) — осуществляет построение соответствий между двумя наборами точек изображений.

ОПРЕДЕЛЕНИЕ

- Особая точка m, или точечная особенность (англ. point feature, key point, feature), изображения – это точка изображения, окрестность которой o(m) можно отличить от окрестности любой другой точки изображения o(n) в некоторой другой окрестности особой точки o2(m).
- В качестве окрестности точки изображения для большинства алгоритмов берётся прямоугольное окно, часто имеющее размер 5x5 пикселей.

ПОДХОДЫ К ОПРЕДЕЛЕНИЮ

- Основанные на интенсивности изображения.
- Использующие контуры изображения: ищут места с максимальным значением кривизны или делают полигональную аппроксимацию контуров и определяют пересечения. Эти методы чувствительны к окрестностям пересечений, поскольку извлечение часто может быть неправильным в тех местах, где пересекаются 3 или более краев.
- На основе использования модели: используются модели с интенсивностью в качестве параметров, которые подстраиваются к изображениям-шаблонам до субпиксельной точности.

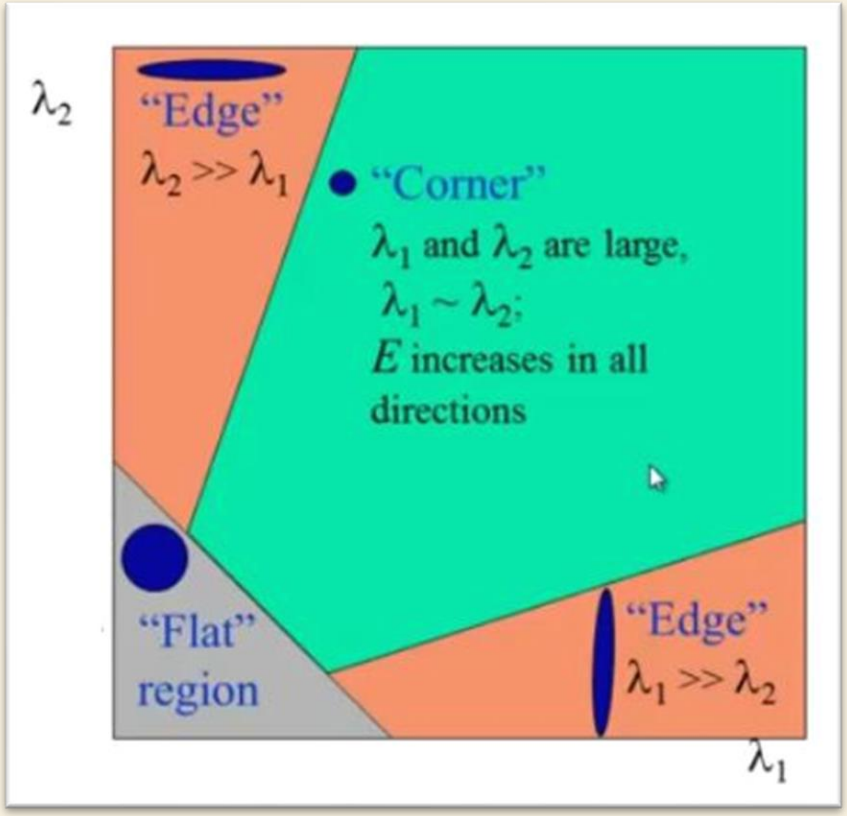
КЛАССИЧЕСКИЙ ПОДХОД – ИЗМЕНЕНИЕ ЯРКОСТИ

- Детектор Харриса
- Вычисляем собственные значения матрицы

$$A^T A = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix}$$

- Вычисляем меру отклика

$$\mathrm{R} = \det A^T A - k \cdot tr A^T A$$



ДЕТЕКТОР ШИ-ТОМАСИ (SHI-TOMASI)

- Аналогичен детектору Харриса, за исключение финального шага. В Shi-Tomasi вычисляется значение

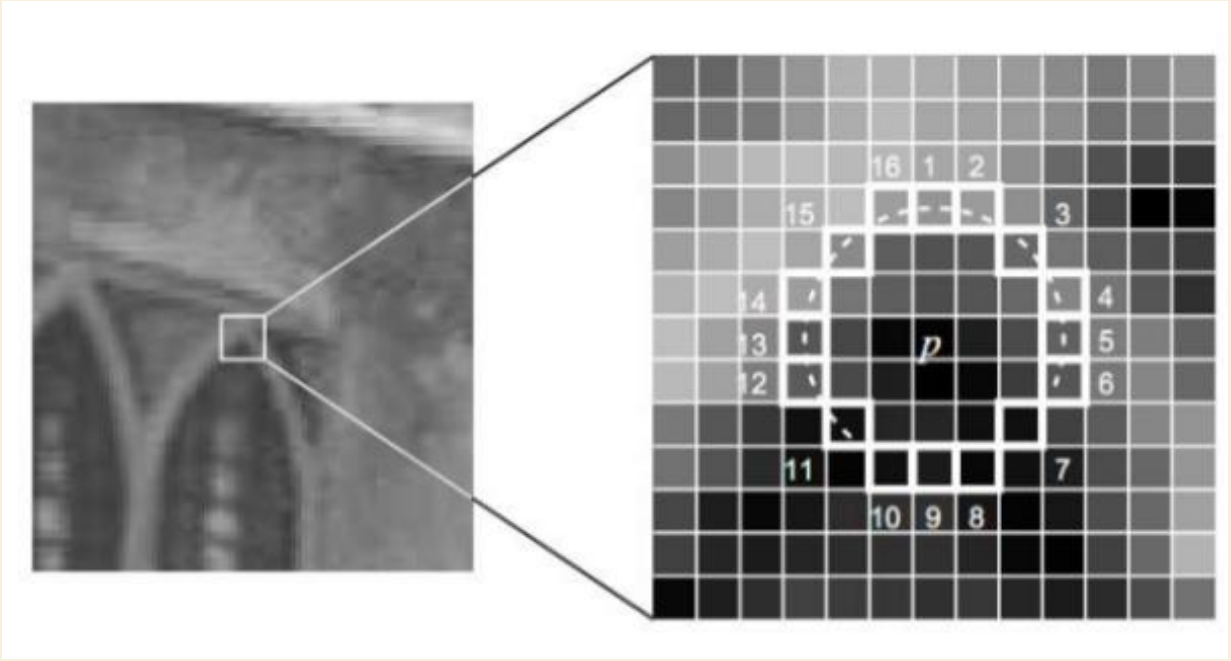
$$R = \min(\lambda_1, \lambda_2)$$

поскольку делается предположение, что поиск углов будет более стабильным.

- В OpenCV реализуется в функции

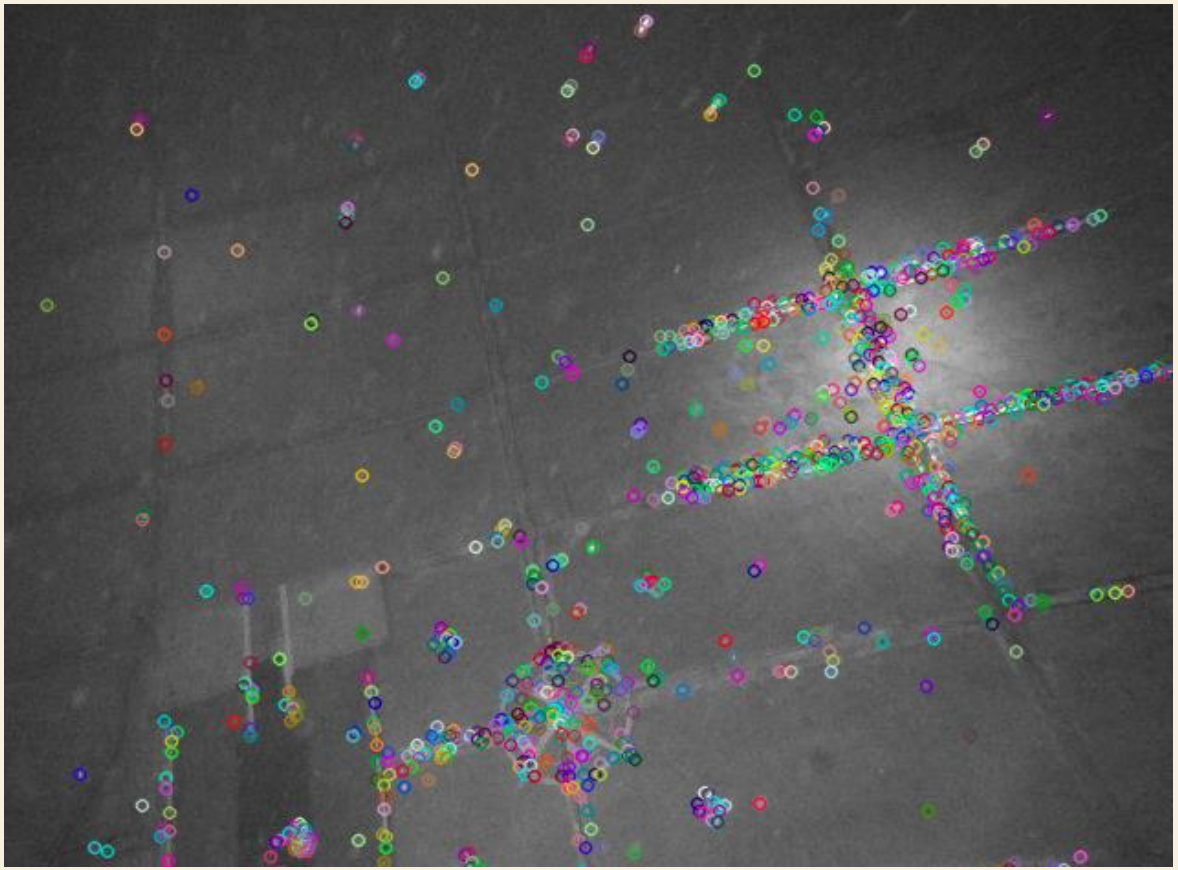
cv::goodFeaturesToTrack()

FAST



- Алгоритм:
- Рассматривается окружность из 16 точек
- Точка считается особой, если на окружности есть N смежных точек, яркость которых больше $I_p + t$ или меньше $I_p - t$
- Оптимизации – проверить точки 1, 5, 9, 13
- Очень быстрый алгоритм

FAST



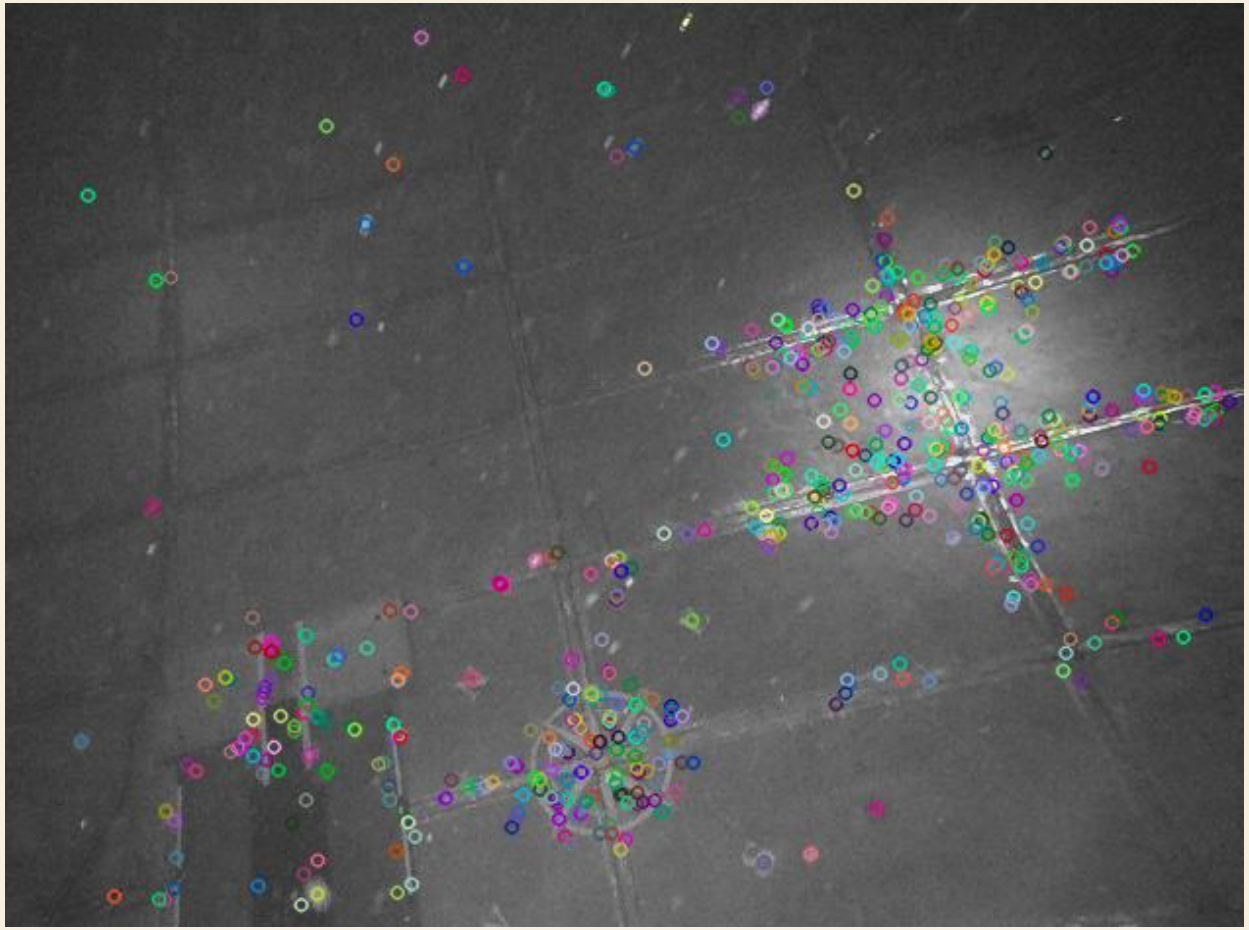
SURF

- Speeded Up Robust Features
- Решает две задачи – поиск особых точек изображения и создание их дескрипторов
- Обнаружение особых точек в SURF основано на вычислении детерминанта матрицы Гессе (гессиана)

$$H\big(f(x,y)\big) = \begin{pmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial^2 f}{\partial x \partial y} \\ \dfrac{\partial^2 f}{\partial x \partial y} & \dfrac{\partial^2 f}{\partial y^2} \end{pmatrix}$$

$$R = \det H = \frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y}\right)^2$$

SURF

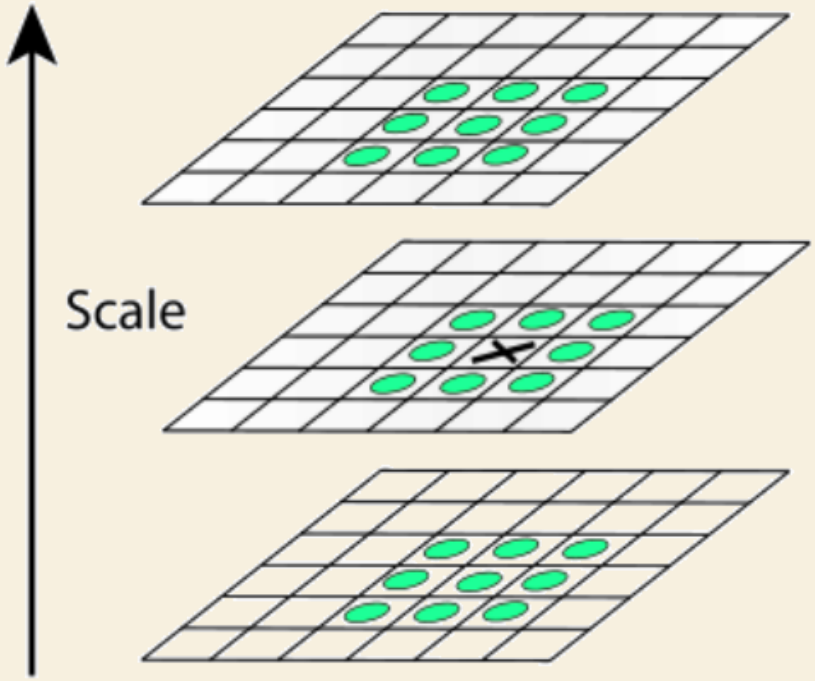


SIFT

- Основным моментом в детектировании особых точек является построение пирамиды гауссианов (Gaussian) и разностей гауссианов (Difference of Gaussian, DoG).
- Гауссианом (или изображением, размытым гауссовым фильтром) является изображение
- Разностью гауссианов называют изображение, полученное путем попиксельного вычитания одного гауссина исходного изображения из гауссиана с другим радиусом размытия.

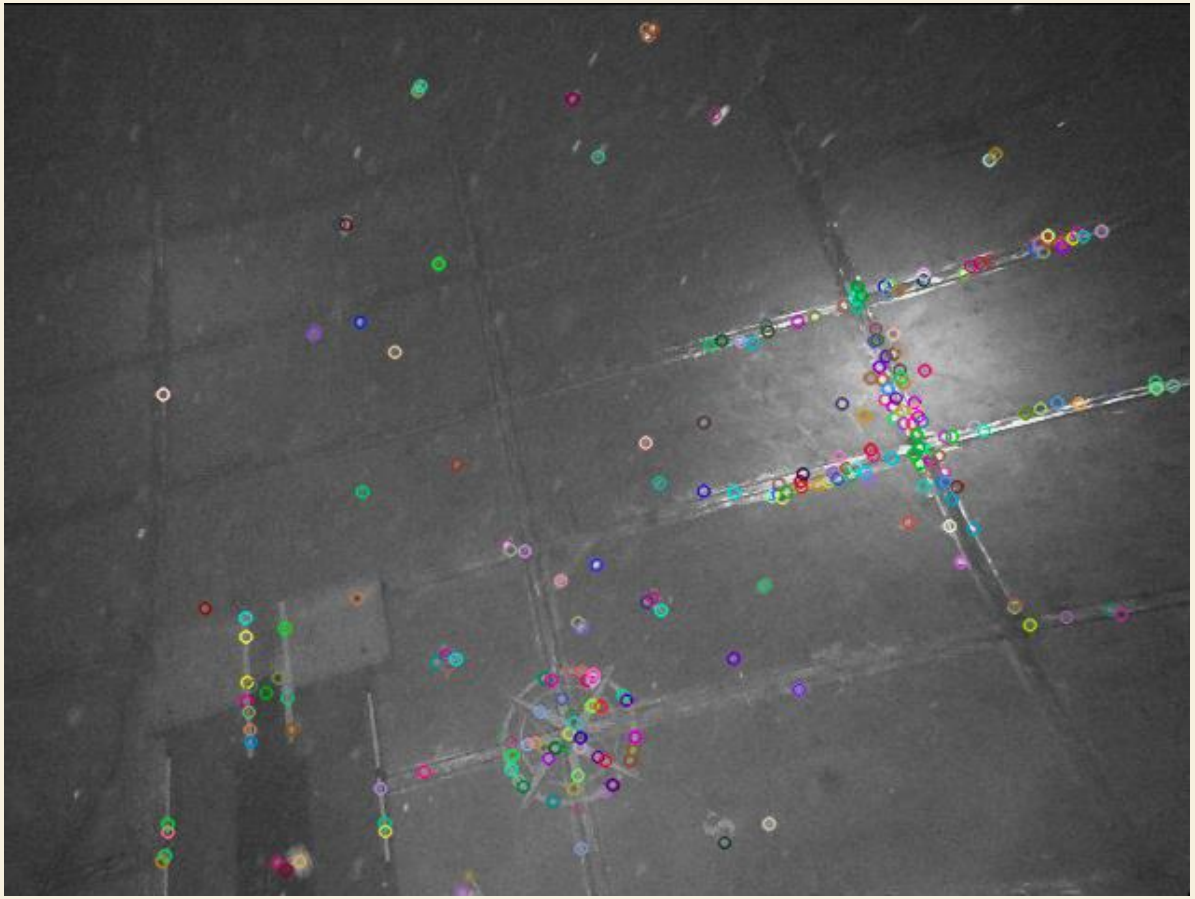
$$D(x, y, \sigma) = [G(x, y, k\sigma) - G(x, y, \sigma)] \approx * I(x, y)$$

SIFT



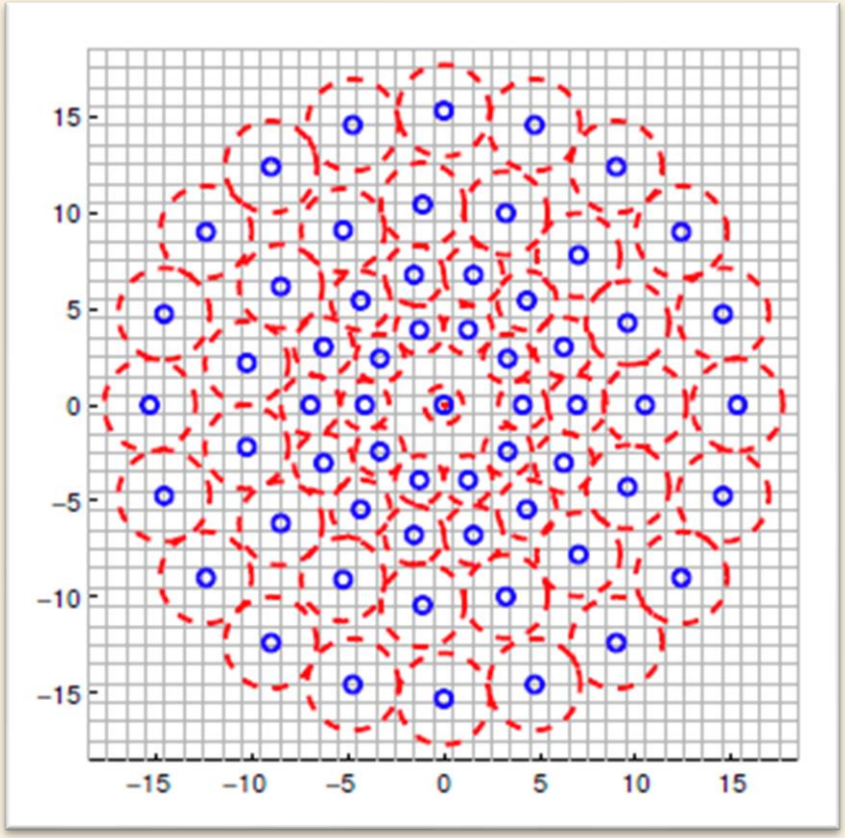
Будем считать точку особой, если она является локальным экстремумом разности гауссианов

SIFT



ТОЧЕЧНЫЙ ДЕТЕКТОР BRISK

- Выделение точечных особенностей - FAST
- Построение их описаний
- Daisy-like дескриптор



```
#include <opencv2/features2d.hpp>
#include <opencv2/xfeatures2d.hpp>
cv::Mat src = cv::imread( "image.jpg", 1 );
cv::Mat imageWithKeypoints;

cv::Ptr<cv::FeatureDetector> detector;
cv::Ptr<cv::DescriptorExtractor> extractor;
cv::BFMatcher matcher;

detector = cv::FastFeatureDetector::create();
detector = cv::xfeatures2d::SURF::create();
detector = cv::xfeatures2d::SIFT::create();

std::vector<cv::KeyPoint> keys1;
detector->detect(src, keys1);

cv::drawKeypoints(src, keys1, imageWithKeypoints);
```

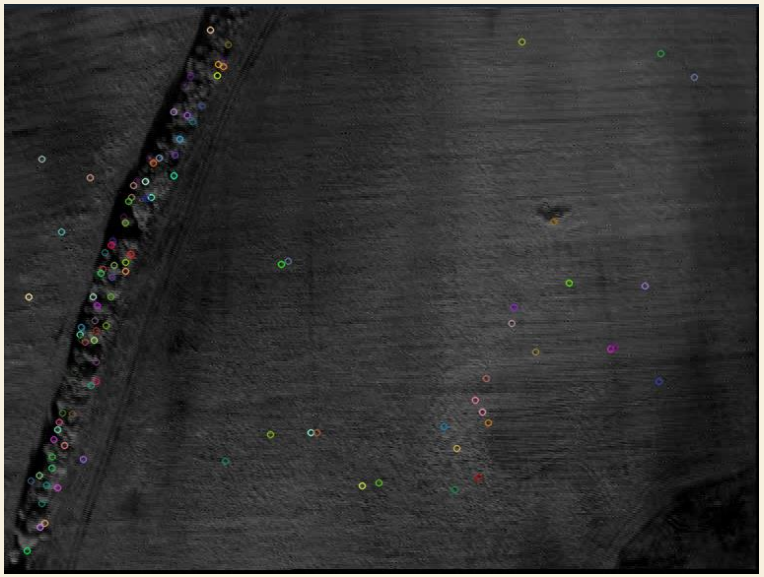
•

```
cv::Mat src;
cv::VideoCapture cap("sample_mpg.avi");
if (!cap.isOpened())
        return -1;

bool stop = false;
// Определим частоту кадров на видео
double rate = cap.get(cv::CAP_PROP_FPS);
// Рассчитаем задержку в миллисекундах
int delay = 1000 / rate;
cout << "Frame rate of video is " << rate << endl;
while (!stop)
{
        // Проверяем доступность кадра
        bool result = cap.grab();
        // Если он готов, считываем
        if (result)
                cap >> src;
        else
                continue;

        cv::imshow("Original", src);
        // Ждём нажатия на кнопку
        int key = cv::waitKey(delay);
        if (key==27) // Если это 0x27, т.е. ESC
                stop=true; // Выходим
}
```

ЗАГРУЗКА ВИДЕО

cv::VideoCapture(filename)

ТОЧЕЧНЫЙ ДЕТЕКТОР BRISK



```
detector = cv::BRISK::create();
extractor = cv::BRISK::create();
detector->detect(src,keys1);
detector->detect(src2,keys2);
cv::Mat descr1, descr2, img_matches;
extractor->compute(src, keys1, descr1);
extractor->compute(src2, keys2, descr2);
std::vector<cv::DMatch> matches;
matcher.match(descr1, descr2, matches);
cv::drawMatches(src, keys1, src2, keys2,\\
 matches, img_matches);
cv::imshow("matches", img_matches);
```

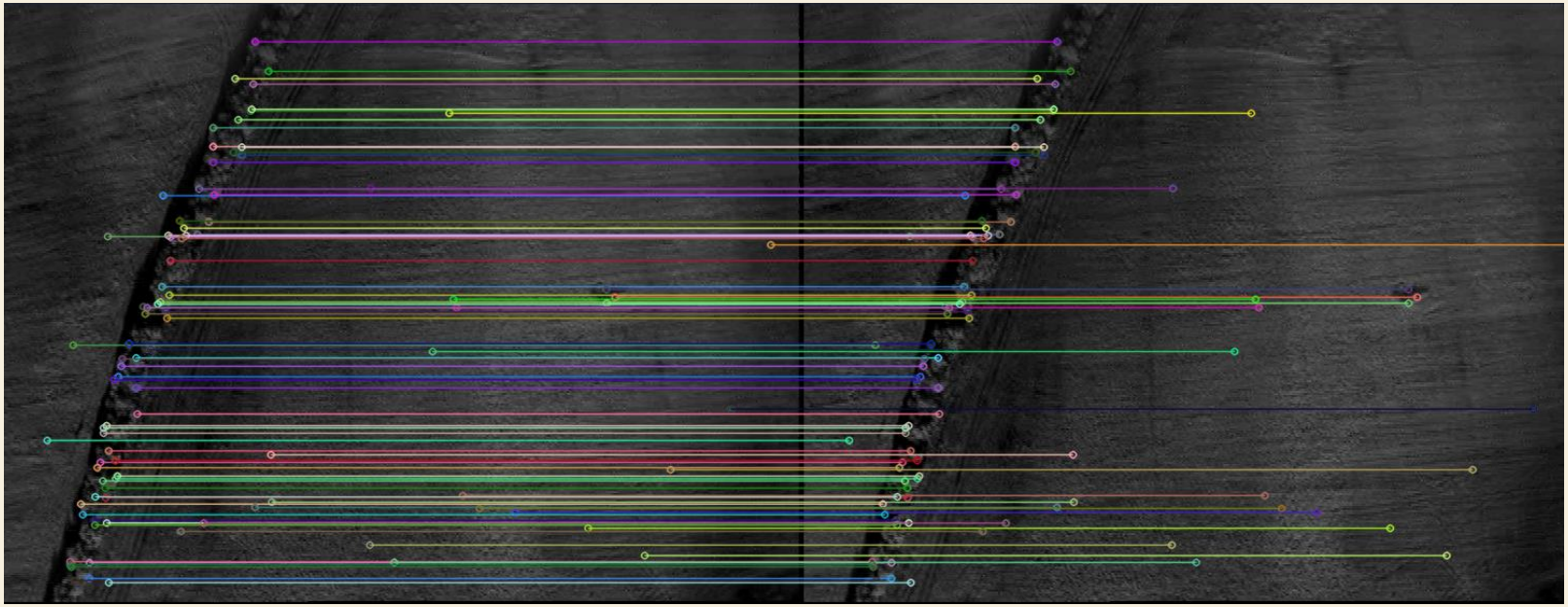
ТОЧЕЧНЫЙ ДЕТЕКТОР BRISK

После выделения точек – можно построить их описание

По описаниям точки можно совместить

Попробуем отобразить их вывод!

ТОЧЕЧНЫЙ ДЕТЕКТОР BRISK



ЧТО НУЖНО ПОДКЛЮЧИТЬ

ЗАГОЛОВКИ

- #include<opencv2/opencv.hpp>
  - Общий заголовок для OpenCV
- #include<opencv2/features2d.hpp>
  - Детекторы FAST, BRISK, ORB
- #include <opencv2/xfeatures2d.hpp>
  - Экспериментальные или лицензированные детекторы (SIFT SURF)

БИБЛИОТЕКИ

- -lopencv_core
  - Основная часть, cv::Mat
- -lopencv_highgui
  - Интерфейс, imshow()
- -lopencv_features2d
  - Детекторы
- -lopencv_xfeatures2d
  - Экспериментальные и лицензированные детекторы
- -lopencv_videoio
  - cv::VideoCapture

ЗАДАЧА 4

- Используйте прилагающийся видеофайл
- Напишите программу для попарного сопоставления кадров:

1. Реализуйте считывание кадров в фреймы
2. Найдите ключевые точки и постройте описания
3. Выведите результат сопоставления на экран
4. Используйте дескрипторы SURF, SIFT и BRISK